Listing of Claims:

Claim 1 (currently amended): A compiler that optimizes a program to be compiled by changing the execution order of instructions in the program in a single phase, the compiler comprising:

an order constraint information obtaining unit that obtains order constraint information indicating order constraints defined among a plurality of instructions in the program, the order constraints defining the order in which the instructions should be executed;

an order determination unit that sequentially determines the execution order for each of the plurality of instructions based on the order constraint information;

a unit for analyzing the number of registers that analyzes the number of required registers, which is the number of registers that will be required when the instructions with its execution order determined among the plurality of instructions are executed;

an instruction detection unit that detects a combination of two instructions, in which one instruction is a determined-order instruction for which the execution order has been determined by the order determination unit, the other instruction is an undetermined-order instruction for which the execution order has not been determined by the order determination unit, and the order constraint information does not include a constraint that the one instruction should be executed before the other instruction; and

an order determination reprocessing unit that, when the number of required registers exceeds a predetermined number, changes the state of the one instruction into the state in which the execution order has not been determined and causes the order determination unit to determine the execution order so that the one instruction is executed next to the other instruction.

Claim 2 (original): The compiler according to claim 1, wherein the instruction detection unit detects an instruction that releases a register as the other instruction, and an instruction that requires a new register allocated to it as the one instruction.

Claim 3 (original): The compiler according to claim 1, wherein the instruction detection unit detects an instruction that releases a register as the other instruction, and an instruction to be

executed before a determined-order instruction that requires a new register allocated to it as the one instruction, and the order determination reprocessing unit further changes the state of all instructions to be executed after the determined-order instruction in the order constraint information into the state in which the execution order has not been determined.

Claim 4 (original): The compiler according to claim 1, wherein when a plurality of combinations of the one instruction and the other instruction are detected by the instruction detection unit, the order determination reprocessing unit selects from the plurality of combinations a combination that minimizes the sum of the depth of order constraint from a start point of the program to the other instruction and the depth of order constraint from the one instruction to an end point of the program, and the order determination reprocessing unit causes the order determination unit to determine the execution order using the other instruction and the one instruction included in the selected combination.

Claim 5 (original): The compiler according to claim 1, wherein when the number of required registers exceeds the predetermined number, the order determination reprocessing unit adds an order constraint that the determined-order instruction should be executed next to the undetermined-order instruction to the order constraint information, and thereby causes the order determination unit to determine the execution order so that the determined-order instruction is executed next to the undetermined-order instruction.

Claim 6 (original): The compiler according to claim 5, wherein the order constraint information obtaining unit obtains, as the order constraint information, an order constraint graph that represents each instruction in the program as a node and order constraints under which a plurality of instructions should be executed as directed edges, the order determination unit determines the execution order based on the order constraint graph so that an instruction represented by a start node of a directed edge is executed before an instruction represented by an end node of the directed edge, the instruction detection unit detects that the order constraint information does not include an order constraint that the one instruction should be executed before the other instruction by detecting a combination of two instructions in which a node

representing the one instruction cannot reach a node representing the other instruction on the order constraint graph, and the order determination reprocessing unit adds an order constraint that the other instruction should be executed next to the one instruction to the order constraint information by generating a directed edge from the node representing the undetermined-order instruction to the node representing the other instruction.

Claim 7 (currently amended): ~~A compiler program~~ A computer program product for causing a computer to function as a compiler that optimizes a program to be compiled by changing the execution order of instructions in the program in a single phase, ~~wherein the compiler program causes the computer to function as:~~ the computer program product comprising a computer usable medium having computer usable program code tangibly embodied therewith, the computer usable medium comprising:

~~an order constraint information obtaining unit that obtains~~ computer usable program code configured to obtain order constraint information indicating order constraints defined among a plurality of instructions in the program, the order constraints defining the order in which the instructions should be executed;

~~an order determination unit that sequentially determines~~ computer usable program code configured to sequentially determine the execution order for each of the plurality of instructions based on the order constraint information;

~~a unit for analyzing the number of registers that analyzes~~ computer usable program code configured to analyze the number of required registers, which is the number of registers that will be required when the instructions with ~~its execution order~~ determined execution order among the plurality of instructions are executed;

~~an instruction detection unit that detects~~ computer usable program code configured to detect a combination of two instructions, in which one instruction is a determined-order instruction for which the execution order has been determined by the order determination unit,

the other instruction is an undetermined-order instruction for which the execution order has not been determined by the order determination unit, and the order constraint information does not include a constraint that the one instruction should be executed before the other instruction; and

~~an order determination reprocessing unit that~~ computer usable program code configured as an order determination reprocessing unit to change, when the number of required registers exceeds a predetermined number, ~~changes~~ the state of the one instruction into the state in which the execution order has not been determined and ~~causes~~ to cause the order determination unit to determine the execution order so that the one instruction is executed next to the other instruction.

Claim 8 (currently amended): The ~~compiler program~~ computer program product according to claim 7, wherein the ~~instruction detection unit~~ computer usable program code configured to detect a combination of two instructions detects an instruction that releases a register as the other instruction, and an instruction that requires a new register allocated to it as the one instruction.

Claim 9 (currently amended): The ~~compiler program~~ computer program product according to claim 7, wherein when the number of required registers exceeds the predetermined number, the ~~order determination reprocessing unit~~ computer usable program code configured to change the state of the one instruction adds an order constraint that the one instruction should be executed next to the other instruction to the order constraint information, ~~and thereby causes the order determination unit to determine the execution order~~ so that the one instruction is executed next to the other instruction.

Claim 10 (canceled)

Claim 11 (canceled)